

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The letters are black and set against a white background. The overall shape is roughly rectangular, with a central vertical column of 'Y's and 'S's. The 'S's are arranged in horizontal rows, while the 'Y's are arranged in vertical columns. The pattern is highly regular and repetitive, creating a complex, textured appearance. The central vertical column is the most prominent feature, with 'Y's at the top and 'S's below. The 'S's are arranged in horizontal rows, with some rows being longer than others. The overall effect is a dense, intricate pattern that resembles a stylized letter or a complex geometric design.

```
IIIIII 000000 SSSSSSSS UU UU BBBB BBBB RRRRRRRR AAAAAA MM MM SSSSSSSS
IIIIII 000000 SS SSSSSSSS UU UU BBBB BBBB RRRRRRRR AAAAAA MM MM SSSSSSSS
II II 00 00 SS SS SS UU UU BB BB RR RR AA AA MMMM MMMM SS
II II 00 00 SS SS SS UU UU BB BB RR RR AA AA MMMM MMMM SS
II II 00 00 SSSSSS UU UU BBBB BBBB RRRRRRRR AA AA MM MM SSSSSS
II II 00 00 SSSSSS SS SS SS UU UU BB BB RR RR AAAAAAAA MM MM SSSSSS
II II 00 00 SS SS SS UU UU BB BB RR RR AAAAAAAA MM MM SS
II II 00 00 SS SS SS UU UU BB BB RR RR AA AA MM MM SS
IIIIII 000000 SSSSSSSS UUUUUUUUU BBBB BBBB RR RR AA AA MM MM SSSSSSSS
IIIIII 000000 SSSSSSSS SSSSSSSS
```

```
LL LL SSSSSSSS
LL LL SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LLLLLLLL SSSSSSSS
LLLLLLLL SSSSSSSS
```

10
Ps
WI

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
74
Th
40
23

Ma
--
--
--
TO
15
Th
MA

IOSUBRAMS
Table of contents

D 10
- NONPAGED RANDOM ACCESS MASS STORAGE I/ 16-SEP-1984 00:25:18 VAX/VMS Macro V04-00

Page 0

(2)	63	APPLY ECC CORRECTION
(3)	126	CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
(4)	173	MAP VIRTUAL TO LOGICAL BLOCK
(5)	319	UPDATE TRANSFER PARAMETERS
(6)	369	SENSE DISK'S SIZE FDT ROUTINE


```
0000 1 .TITLE IOSUBRAMS - NONPAGED RANDOM ACCESS MASS STORAGE I/O RELATED ROUTINES
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28 AUTHOR: D. N. CUTLER 16-MAR-1977
0000 29
0000 30 MODIFIED BY:
0000 31
0000 32 V02-005 ACG0237 Andrew C. Goldstein, 9-Dec-1981 11:39
0000 33 Add cathedral window support; check mapping against
0000 34 file size in FCB
0000 35
0000 36 V02-004 GRR2004 Greg R. Robert, 14-Jun-1981
0000 37 Added alternate entry point to IOC$CVTLOGPHY
0000 38
0000 39 V02-003 ACG0176 Andrew C. Goldstein, 6-Jun-1980 13:54
0000 40 Redirect UCB to file's UCB on unsuccessful virtual map
0000 41
0000 42 **
0000 43
0000 44
0000 45 NONPAGED RANDOM ACCESS MASS STORAGE I/O RELATED ROUTINES
0000 46
0000 47 MACRO LIBRARY CALLS
0000 48
0000 49
0000 50 $DEVDEF ;DEFINE DEVICE CHARACTERISTIC BITS
0000 51 $DYNDEF ;DEFINE DATA STRUCTURE TYPE CODES
0000 52 $FCBDEF ;DEFINE FCB OFFSETS
0000 53 $IPLDEF ;DEFINE INTERRUPT PRIORITY LEVELS
0000 54 $IRPDEF ;DEFINE IRP OFFSETS
0000 55 $PRDEF ;DEFINE PROCESSOR REGISTERS
0000 56 $PTEDEF ;DEFINE PAGE TABLE ENTRY FIELDS
0000 57 $RVDEF ;DEFINE RVT OFFSETS
```


IOSUBRAMS
V04-000

- NONPAGED RANDOM ACCESS MASS STORAGE I/ 16-SEP-1984 00:25:18 VAX/VMS Macro V04-00
5-SEP-1984 03:43:47 [SYS.SRC]IOSUBRAMS.MAR;1

Page 2
(1)

0000	58	\$SSDEF
0000	59	\$UCBDEF
0000	60	\$VADEF
0000	61	\$WCBDEF

;DEFINE I/O STATUS CODES
;DEFINE UCB OFFSETS
;DEFINE VIRTUAL ADDRESS FIELDS
;DEFINE WCB OFFSETS

```
0000 63 .SBTTL APPLY ECC CORRECTION
0000 64 :+
0000 65 : IOC$APPLYECC - APPLY ECC CORRECTION
0000 66 :
0000 67 : THIS ROUTINE IS CALLED TO APPLY AN ECC CORRECTION TO DATA THAT HAS BEEN
0000 68 : TRANSFERED INTO MEMORY FROM A DISK DEVICE.
0000 69 :
0000 70 : INPUTS:
0000 71 :
0000 72 : R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED UP TO, BUT NOT
0000 73 : INCLUDING, BLOCK TO BE CORRECTED (MUST BE A MULTIPLE OF 512
0000 74 : BYTES).
0000 75 : R5 = DEVICE UNIT UCB ADDRESS.
0000 76 :
0000 77 : UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
0000 78 : UCB$W_EC1(R5) = STARTING BIT NUMBER OF ERROR BURST.
0000 79 : UCB$W_EC2(R5) = EXCLUSIVE OR CORRECTION PATTERN.
0000 80 : UCB$S_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
0000 81 : THE TRANSFER.
0000 82 :
0000 83 : OUTPUTS:
0000 84 :
0000 85 : THE CORRECTION PATTERN IS EXCLUSIVE OR'ED WITH THE DATA IN MEMORY
0000 86 : PROVIDING THE NECESSARY CORRECTION.
0000 87 :
0000 88 : R3 IS PRESERVED ACROSS CALL.
0000 89 :-
0000 90
0000 91 .PSECT WIONONPAGED
0000 92 IOC$APPLYECC::
0000 93 PUSH R3,R4
0000 94 MOVZWL UCB$W_EC1(R5),R2
0000 95 DECL R2
0000 96 BICB3 #XF8,R2,R1
0000 97 DIVL #8,R2
0000 98 ADDL R0,R2
0000 99 MOVZWL UCB$W_EC2(R5),R0
0000 100 ASHL R1,R0,R0
0000 101 MOVL #3,R4
0000 102 10$: CMPW R2,UCB$W_BCNT(R5)
0000 103 BGEQU 40$
0000 104 MOVZWL UCB$W_BOFF(R5),R1
0000 105 ADDL R2,R1
0000 106 ASHL #-VASS_BYTE,R1,R1
0000 107 MOVL @UCB$S_SVAPTE(R5)[R1],R3
0000 108 BLSS 20$
0000 109 BSBW IOC$PTETOPFN
0000 110 20$: MULL3 #4,UCB$S_SVPN(R5),R1
0000 111 INSV R3,#PTE$V_PFN,-
0000 112 #PTE$S_PFN,@MMG$GL_SPTBASE[R1]
0000 113 ASHL #VASS_BYTE-2,R1,R1
0000 114 BBSS #VASS_SYSTEM,R1,30$
0000 115 30$: INVALID R1
0000 116 ADDL3 UCB$W_BOFF(R5),R2,R3
0000 117 INSV R3,#VASS_BYTE,#VASS_BYTE,R1
0000 118 XORB R0,(R1)
0000 119 ASHL #-8,R0,R0
0000 120 : APPLY ECC CORRECTION
0000 121 : SAVE REGISTERS
0000 122 : GET STARTING BIT NUMBER OF ERROR BURST
0000 123 : CONVERT TO RELATIVE BIT NUMBER
0000 124 : ISOLATE PATTERN SHIFT COUNT
0000 125 : CALCULATE RELATIVE BYTE OFFSET IN BLOCK
0000 126 : CALCULATE RELATIVE OFFSET IN BUFFER
0000 127 : GET EXCLUSIVE OR CORRECTION PATTERN
0000 128 : SHIFT PATTERN TO PROPER POSITION
0000 129 : SET LOOP COUNT
0000 130 : BYTE OFFSET WITHIN RANGE?
0000 131 : IF GEQU NO
0000 132 : GET BYTE OFFSET IN PAGE
0000 133 : CALCULATE BYTE OFFSET OF TRANSFER PTE
0000 134 : CALCULATE LONGWORD OFFSET TO TRANSFER PTE
0000 135 : GET TRANSFER PTE
0000 136 : IF LSS VALID PTE
0000 137 : CONVERT TO VALID PTE
0000 138 : CALCULATE BYTE OFFSET TO SYSTEM PTE
0000 139 : MOVE TRANSFER PTE INTO SYSTEM PAGE TABLE
0000 140 :
0000 141 : CONVERT SVPN TO SYSTEM VIRTUAL ADDRESS
0000 142 : SET SYSTEM VIRTUAL ADDRESS BIT
0000 143 : INVALIDATE TRANSLATION BUFFER
0000 144 : CALCULATE BYTE OFFSET IN BLOCK
0000 145 : INSERT BYTE OFFSET IN BLOCK
0000 146 : CORRECT MEMORY BYTE
0000 147 : SHIFT NEXT CORRECTION BYTE INTO PLACE
```

52	00C4	18	BB	0000	93
51	52	F8	8F	8B	0009
	52	08	C6	000E	97
	52	50	C0	0011	98
50	00C6	C5	3C	0014	99
50	50	51	78	0019	100
	54	03	D0	001D	101
7E	A5	52	B1	0020	102
		47	1E	0024	103
51	7C	A5	3C	0026	104
	51	52	C0	002A	105
51	51	F7	8F	78	002D
53	78	B5	41	D0	0032
		03	19	0037	108
	FFC4		30	0039	109
51	74	A5	C5	003C	110
	00	53	F0	0041	111
00000000	FF41	15		0044	112
51	51	07	78	004B	113
	00	51	E2	004F	114
		1F		0053	115
53	52	7C	A5	C1	0056
51	09	00	53	F0	005B
	61	50	8C	0060	118
50	50	F8	8F	78	0063

IOSUBRAMS
V04-000

- NONPAGED RANDOM ACCESS MASS STORAGE I/ H 10
APPLY ECC CORRECTION 16-SEP-1984 00:25:18 VAX/VMS Macro V04-00
5-SEP-1984 03:43:47 [SYS.SRC]IOSUBRAMS.MAR;1

Page 4
(2)

68	A5	01	D6	0068	120	INCL	R2		
			F5	006A	121	SOBGTR	R4,10\$		;UPDATE OFFSET IN BUFFER
			BA	006D	122	POPR	#^M<R3,R4>		;ANY MORE CORRECTIONS TO MAKE?
			A8	006F	123	BISW	#UCB\$M_ECC,UCB\$W_DEVSTS(R5)		;RESTORE REGISTERS
			05	0073	124	RSB			;SET ECC CORRECTION MADE

40\$:

```
0074 126 .SBTTL CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
0074 127 :+
0074 128 : IOC$CVTLOGPHY - CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
0074 129 :
0074 130 : THIS ROUTINE IS CALLED TO CONDITIONALLY CONVERT A LOGICAL BLOCK NUMBER
0074 131 : TO A PHYSICAL DISK ADDRESS AND STORE THE RESULT IN THE I/O PACKET.
0074 132 :
0074 133 : INPUTS:
0074 134 :
0074 135 : R0 = LOGICAL BLOCK NUMBER TO BE CONVERTED.
0074 136 : R3 = I/O PACKET ADDRESS.
0074 137 : R5 = DEVICE UNIT UCB ADDRESS.
0074 138 :
0074 139 : OUTPUTS:
0074 140 :
0074 141 : IF UCB$V_NOCNVRT IS CLEAR IN UCB$W_DEVSTS, THE LOGICAL BLOCK NUMBER
0074 142 : IS CONVERTED TO A PHYSICAL DISK ADDRESS USING THE DISK GEOMETRY PARA-
0074 143 : METERS IN THE UCB. THE RESULT IS STORED IN THE MEDIA ADDRESS LONGWORD
0074 144 : OF THE I/O PACKET.
0074 145 :
0074 146 : IF UCB$V_NOCNVRT IS SET, THE BLOCK NUMBER IS STORED IN THE MEDIA ADDRESS
0074 147 : LONGWORD WITHOUT CONVERSION.
0074 148 :
0074 149 : IF THE ROUTINE IS ENTERED AT IOC$CVTLOGPHYU, THEN UCB$V_NOCNVRT IS
0074 150 : IGNORED.
0074 151 :
0074 152 : R3 IS PRESERVED ACROSS CALL.
0074 153 :-
0074 154 :
0074 155 : .ENABLE LOCAL_BLOCK
0074 156 :
0074 157 IOC$CVTLOGPHY:: :CONVERT LOGICAL BLOCK TO PHYSICAL ADDRESS
0074 158 MOVL R0,IRP$L_MEDIA(R3) :ASSUME NO CONVERSION
0078 159 BBS #UCB$V_NOCNVRT,UCB$W_DEVSTS(R5),10$ :BYPASS CONVERSION IF SET
007D 160 IOC$CVTLOGPHYU:: :UNCONDITIONAL ENTRY POINT
007D 161 MOVZBL UCB$L_DEVDEPEND(R5),R2 :GET NUMBER OF SECTORS PER TRACK
0081 162 CLRL R1 :CLEAR HIGH PART OF DIVIDEND
0083 163 EDIV R2,R0,R0,IRP$L_MEDIA(R3) :CALCULATE SECTOR NUMBER AND STORE
0089 164 MOVZBL UCB$L_DEVDEPEND+1(R5),R2 :GET NUMBER OF TRACKS PER CYLINDER
008D 165 EDIV R2,R0,R0,R1 :CALCULATE TRACK AND CYLINDER
0092 166 MOV B R1,IRP$L_MEDIA+1(R3) :STORE TRACK NUMBER
0096 167 MOVW R0,IRP$L_MEDIA+2(R3) :STORE CYLINDER NUMBER
009A 168 10$: RSB :
009B 169 :
009B 170 :
009B 171 : .DISABLE LOCAL_BLOCK
```

38 A3	50	D0
1D 68 A5	02	E0
52 44 A5	9A	007D
51	D4	0081
38 A3	50 50 52	7B 0083
51	52 45 A5	9A 0089
50	50 52	7B 008D
39 A3	51	90 0092
3A A3	50	B0 0096
		05 009A
		009B
		009B
		009B


```
009B 173 .SBTTL MAP VIRTUAL TO LOGICAL BLOCK
009B 174 :+
009B 175 : IOC$MAPVBLK - MAP VIRTUAL TO LOGICAL BLOCK
009B 176 :
009B 177 : THIS ROUTINE IS CALLED TO MAP A VIRTUAL BLOCK TO A LOGICAL BLOCK USING A
009B 178 : MAPPING WINDOW.
009B 179 :
009B 180 : INPUTS:
009B 181 :
009B 182 : R0 = VIRTUAL BLOCK NUMBER.
009B 183 : R1 = NUMBER OF BYTES TO MAP.
009B 184 : R2 = ADDRESS OF WINDOW MAPPING BLOCK.
009B 185 : R5 = UCB ADDRESS OF DEVICE UNIT.
009B 186 :
009B 187 : OUTPUTS:
009B 188 :
009B 189 : R0 LOW BIT CLEAR INDICATES A TOTAL MAPPING FAILURE.
009B 190 :
009B 191 : R2 = NUMBER OF UNMAPPED BYTES.
009B 192 :
009B 193 : R0 LOW BIT SET INDICATES PARTIAL MAP WITH:
009B 194 :
009B 195 : R1 = LOGICAL BLOCK NUMBER OF FIRST BLOCK.
009B 196 : R2 = NUMBER OF UNMAPPED BYTES.
009B 197 : R5 = UCB ADDRESS OF DEVICE UNIT (POSSIBLY MODIFIED).
009B 198 :
009B 199 : R3 IS PRESERVED ACROSS CALL.
009B 200 : -
009B 201 :
009B 202 : .ENABLE LOCAL_BLOCK
009B 203 :
009B 204 10$: BUG_CHECK STRNOTWCB,FATAL ;STRUCTURE IS NOT A WINDOW BLOCK
009F 205 :
009F 206 IOC$MAPVBLK:: ;MAP VIRTUAL TO LOGICAL BLOCK
009F 207 20$: DSBINT UCB$B_FIPL(R5) ;SYNCHRONIZE ACCESS TO SYSTEM DATABASE
009F 208 : PUSHF #M<R0,R1,R2,R3,R4,R5> ;SAVE REGISTERS
009F 209 : MOVAB -1(R0),R7 ;SAVE START VBN -1
009F 210 : MOVL R2,R3 ;GET COPY OF WINDOW ADDRESS
00B1 211 :
00B1 212 : THE WINDOW MAY CONSIST OF A CHAIN OF WCB SEGMENTS. SEARCH THROUGH THE
00B1 213 : CHAIN UNTIL WE FIND ONE WHICH IS BEYOND THE DESIRED VBN OR WE REACH
00B1 214 : THE END OF THE CHAIN.
00B1 215 :
00B1 216 30$: CMPB WCB$B_TYPE(R3),#DYN$C_WCB ;SEE IF THIS IS REALLY A WINDOW
00B1 217 : BNEQ 10$ ;IF NEQ NO
00B1 218 : CMPL R0,WCB$B_STVBN(R3) ;CHECK VBN AGAINST START VBN OF WINDOW
00B1 219 : BLSSU 40$ ;BRANCH IF VBN PRECEDES WINDOW
00B1 220 : MOVL R3,R2 ;ELSE ADVANCE TO THIS WINDOW SEGMENT
00B1 221 : MOVL WCB$B_LINK(R2),R3 ;LOOK AT NEXT WINDOW SEGMENT
00B1 222 : BNEQ 30$ ;BRANCH TO LOOK AT IF IT EXISTS
00B1 223 :
00B1 224 40$: MOVZWL WCB$B_NMAP(R2),R3 ;GET COUNT OF RETRIEVAL POINTERS
00B1 225 : BEQL 60$ ;BRANCH IF EMPTY WINDOW
00B1 226 : MOVAL WCB$B_STVBN(R2),R4 ;POINT TO STARTING VBN
00B1 227 : SUBL (R4)+,R0 ;SUBTRACT STARTING VBN FROM DESIRED
00B1 228 : BBS #DEV$V_SQD,UCB$B_DEVCHAR(R5),70$ ;IF SET, SEQUENTIAL DEVICE
00B1 229 : BLSSU 60$ ;BRANCH IF VBN PRECEDES WINDOW

57 01DA 8F BB 00A6 208
FF A0 9E 00AA 209
53 52 D0 00AE 210
00B1 211
00B1 212
00B1 213
00B1 214
00B1 215
12 0A A3 91 00B1 216 30$:
E4 12 00B5 217
2C A3 50 D1 00B7 218
09 1F 00BB 219
52 53 D0 00BD 220
20 A2 D0 00C0 221
EB 12 00C4 222
00C6 223
53 16 A2 3C 00C6 224 40$:
1B 13 00CA 225
54 2C A2 DE 00CC 226
50 84 C2 00D0 227
17 38 A5 05 E0 00D3 228
OD 1F 00D8 229
```

```
00DA 230
00DA 231 :
00DA 232 : SCAN THE WINDOW, SUBTRACTING THE COUNT FIELD OF EACH POINTER FROM THE
00DA 233 : CURRENT RELATIVE BLOCK NUMBER.
00DA 234 :
00DA 235 :
51 84 3C 00DA 236 50$: MOVZWL (R4)+,R1 ;GET COUNT FIELD OF RETRIEVAL POINTER
50 51 C2 00DD 237 : SUBTRACT FROM RELATIVE BLOCK NUMBER
12 1F 00E0 238 : BRANCH IF VBN LOCATED IN THIS POINTER
84 D5 00E2 239 : SKIP LBN FIELD OF POINTER
F3 53 F5 00E4 240 : LOOP THRU WINDOW
50 D4 00E7 241 60$: CLRL R0 ;VBN IS BEYOND WINDOW
55 10 A2 D0 00E9 242 : REDIRECT UCB TO VOLUME CONTAINING THE FILE
5F 11 00ED 243 : RETURN FAILURE
00EF 244 :
00EF 245 :
00EF 246 : DEVICE IS A SEQUENTIAL DEVICE. FIRST MAPPING POINTER CONTAINS THE UCB ADDRESS
00EF 247 : OF THE CURRENT VOLUME THAT IS BEING PROCESSED. ALL BYTES ALWAYS MAP.
00EF 248 :
00EF 249 :
55 64 D0 00EF 250 70$: MOVL (R4),R5 ;GET CURRENT VOLUME UCB ADDRESS
55 55 11 00F2 251 :
00F4 252 :
00F4 253 :
00F4 254 : FOUND THE RETRIEVAL POINTER CONTAINING THE STARTING VBN. R0 NOW
00F4 255 : CONTAINS A NEGATIVE VALUE WHICH IS THE NUMBER OF BLOCKS BETWEEN
00F4 256 : THE STARTING VBN AND THE END OF THE POINTER.
00F4 257 :
00F4 258 :
58 50 D0 00F4 259 80$: MOVL R0,R8 ;SAVE # BLOCKS MAPPED PAST START VBN
51 84 C0 00F7 260 : FIRST LBN BEYOND THIS POINTER
50 51 C0 00FA 261 : COMPUTE STARTING LBN
00FD 262 :
00FD 263 :
00FD 264 : IF THE NEXT RETRIEVAL POINTER IS CONTIGUOUS WITH THE ONE FOUND, ADD
00FD 265 : IN ITS COUNT TO HANDLE THE CASE WHERE A TRANSFER SPANS TWO POINTERS.
00FD 266 : NOTE THAT THE GREATEST NUMBER OF CONTIGUOUS POINTERS A TRANSFER CAN
00FD 267 : SPAN IS TWO.
00FD 268 :
00FD 269 :
53 53 D7 00FD 270 : DECL R3 ;SEE IF THERE IS ANOTHER POINTER
08 15 00FF 271 : BLEQ 90$ ;BRANCH IF NONE
53 84 3C 0101 272 : MOVZWL (R4)+,R3 ;GET COUNT OF NEXT RETRIEVAL POINTER
64 51 D1 0104 273 : CMPL R1,(R4) ;SEE IF THE NEXT POINTER IS CONTIGUOUS
03 12 0107 274 : BNEQ 90$ ;BRANCH IF NOT
58 53 C2 0109 275 : SUBL R3,R8 ;ADD TO # BLOCKS MAPPED (NEGATIVE)
010C 276 :
010C 277 :
010C 278 : EXTRACT THE LBN AND RVN COMPONENTS OF THE STARTING 'LBN' AND SWITCH
010C 279 : TO THE RIGHT UCB IF THIS IS A MULTI-VOLUME SET.
010C 280 :
010C 281 :
51 50 18 00 51 50 18 00 EF 010C 282 90$: EXTZV #0,#24,R0,R1 ;EXTRACT LBN PART
50 50 08 18 EF 0111 283 : EXTZV #24,#8,R0,R0 ;EXTRACT RVN
09 13 0116 284 : BEQL 100$ ;BRANCH IF NOT VOLUME SET
56 1C A2 D0 0118 285 : MOVL WCB$L_RVT(R2),R6 ;GET RELATIVE VOLUME TABLE ADDR
55 40 A640 D0 011C 286 : MOVL RVT$L_UCBLST-4(R6)(R0),R5 ;GET THE RIGHT UCB ADDRESS
```



```
0121 287
0121 288 :
0121 289 : CHECK THE RANGE OF VBN'S PROVIDED BY THE MAP POINTER AGAINST THE
0121 290 : FILE SIZE RECORDED IN THE FCB. REDUCE IT IF THE FCB INDICATES A
0121 291 : SMALLER FILE SIZE THAN THE WINDOW.
0121 292 :
0121 293 :
12 0B A2 02 E0 0121 294 100$: BBS #WCB$V_NOTFCP,WCB$B_ACCESS(R2),110$ :SKIP CHECK IF NO FCB
56 18 A2 D0 0126 295 :MOVL WCB$L_FCB(R2),R6 :GET FCB ADDRESS
57 38 A6 C2 012A 296 :SUBL FCB$L_FILESIZE(R6),R7 :COMPUTE NEG BLOCKS PAST DESIRED VBN
58 57 B7 1E 012E 297 :BGEQU 60$ :BRANCH IF VBN PAST END OF FILE
58 03 1F 0130 298 :CMPL R7,R8 :COMPARE AGAINST BLOCKS MAPPED
58 57 D0 0133 299 :BLSSU 110$ :BRANCH IF LESS MAPPED BY WINDOW
0135 300 :MOVL R7,R8 :ELSE LIMIT TO FILE SIZE
0138 301 :
0138 302 :
0138 303 : SEE IF THE ENTIRE TRANSFER IS MAPPED CONTIGUOUSLY.
0138 304 :
0138 305 :
00000000'EF D6 0138 306 110$: INCL PMSS$GL_HIT :COUNT A WINDOW HIT
58 58 09 78 013E 307 :ASHL #9,R8,R8 :CONVERT TO # BYTES MAPPED
6E 05 1D 0142 308 :BVS 120$ :BRANCH IF COUNT IS HUGE
6E 58 C0 0144 309 :ADDL R8,(SP) :SUBTRACT FROM BYTES DESIRED
50 02 18 0147 310 :BGEQ 130$ :BRANCH IF NOT TOTAL MAP
01DC 8F D4 0149 311 120$: CLRL (SP) :ZERO INDICATES COMPLETE MAP
D0 014B 312 130$: MOVL #SS$ NORMAL,R0 :INDICATE SUCCESS
BA 014E 313 140$: POPR #^M<R2,R3,R4,R6,R7,R8> :RESTORE REGISTERS
0152 314 :ENBINT :ALLOW INTERRUPTS
05 0155 315 :RSB :
0156 316 :
0156 317 :.DISABLE LOCAL_BLOCK
```

```
0156 319 .SBTTL UPDATE TRANSFER PARAMETERS
0156 320 :+
0156 321 : IOCSUPDATRANS - UPDATE TRANSFER PARAMETERS
0156 322 :
0156 323 : THIS ROUTINE IS CALLED TO UPDATE THE TRANSFER PARAMETERS AFTER A DISK ERROR
0156 324 : HAS BEEN DISCOVERED BUT GOOD DATA WAS TRANSFERRED.
0156 325 :
0156 326 : INPUTS:
0156 327 :
0156 328 : R0 = NUMBER OF BYTES OF DATA THAT WERE TRANSFERED (MUST BE A MULTIPLE
0156 329 : OF 512 BYTES).
0156 330 : R5 = DEVICE UNIT UCB ADDRESS.
0156 331 :
0156 332 : UCB$W_BCNT(R5) = LENGTH OF TRANSFER IN BYTES.
0156 333 : UCB$W_DA(R5) = CURRENT SECTOR AND TRACK ADDRESS.
0156 334 : UCB$W_DC(R5) = CURRENT CYLINDER ADDRESS.
0156 335 : UCB$L_SVAPTE(R5) = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE THAT MAPS
0156 336 : THE TRANSFER.
0156 337 :
0156 338 : OUTPUTS:
0156 339 :
0156 340 : THE NUMBER OF BYTES REMAINING TO BE TRANSFERED, THE SYSTEM VIRTUAL
0156 341 : ADDRESS OF THE NEXT PTE, AND THE CURRENT DISK ADDRESS OF THE TRANSFER
0156 342 : ARE UPDATED.
0156 343 :
0156 344 : R3 IS PRESERVED ACROSS CALL.
0156 345 :-
0156 346 :
0156 347 IOCSUPDATRANS::
0156 348 SUBW R0,UCB$W_BCNT(R5) ;UPDATE TRANSFER PARAMETERS
0156 349 ASHL #-7,R0,R0 ;CALCULATE REMAINING BYTES TO TRANSFER
0156 350 ADDL R0,UCB$L_SVAPTE(R5) ;CALCULATE PTE LONGWORDS TO SKIP OVER
0156 351 DIVL #4,R0 ;UPDATE SYSTEM VIRTUAL ADDRESS OF NEXT PTE
0156 352 ADDB R0,UCB$W_DA(R5) ;CALCULATE NUMBER OF SECTORS TRANSFERED
0156 353 ;UPDATE SECTOR ADDRESS
0156 354 :
0156 355 : RIPPLE CARRY FROM SECTOR TO TRACK AND FROM TRACK TO CYLINDER
0156 356 :
0156 357 :
0156 358 10$: CMPB UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SECTOR OVERFLOW?
0156 359 BGTRU 20$ ;IF GTRU NO
0156 360 SUBB UCB$L_DEVDEPEND(R5),UCB$W_DA(R5) ;SUBTRACT OUT A TRACK
0156 361 INCB UCB$W_DA+1(R5) ;INCREMENT TRACK ADDRESS
0156 362 CMPB UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;TRACK OVERFLOW?
0156 363 BGTRU 10$ ;IF GTRU NO
0156 364 SUBB UCB$L_DEVDEPEND+1(R5),UCB$W_DA+1(R5) ;SUBTRACT OUT A CYLINDER
0156 365 INCW UCB$W_DC(R5) ;UPDATE CYLINDER ADDRESS
0156 366 BRB 10$
0156 367 20$: RSB
```

50 7E A5 50 A2 0156 348
50 50 F9 8F 78 015A 349
78 A5 50 C0 015F 350
50 04 C6 0163 351
00BC C5 50 80 0166 352
0168 353
0168 354
0168 355
0168 356
0168 357
00BC C5 44 A5 91 016B 358
1E 1A 0171 359
00BC C5 44 A5 82 0173 360
00BD C5 00BD C5 96 0179 361
45 A5 91 017D 362
E6 1A 0183 363
00BD C5 45 A5 82 0185 364
00BE C5 B6 018B 365
DA 11 018F 366
05 0191 367


```
0192 369 .SBTTL SENSE DISK'S SIZE FDT ROUTINE
0192 370 :+
0192 371 : IOC$SENSEDISK - SENSE DISK'S SIZE FDT ROUTINE
0192 372 :
0192 373 : THIS ROUTINE IS THE STANDARD SENSEMODE/SENSECHAR FDT ROUTINE FOR
0192 374 : DISK DEVICES. IT OBTAINS THE DISK'S SIZE, IN LOGICAL BLOCKS, FROM THE
0192 375 : UCB (UCB$L_MAXBLOCK) AND IMMEDIATELY COMPLETES THE I/O REQUEST WITH
0192 376 : THE SECOND LONGWORD OF THE FINAL I/O STATUS EQUAL TO THE DISK'S SIZE.
0192 377 :
0192 378 : INPUTS:
0192 379 :
0192 380 : R0 = SCRATCH.
0192 381 : R1 = SCRATCH.
0192 382 : R2 = SCRATCH.
0192 383 : R3 = ADDRESS OF I/O REQUEST PACKET.
0192 384 : R4 = CURRENT PROCESS PCB ADDRESS.
0192 385 : R5 = ASSIGNED DEVICE UCB ADDRESS.
0192 386 : R6 = ADDRESS OF CCB.
0192 387 : R7 = I/O FUNCTION CODE BIT NUMBER.
0192 388 : R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
0192 389 : R9 = SCRATCH.
0192 390 : R10 = SCRATCH.
0192 391 : R11 = SCRATCH.
0192 392 : AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
0192 393 :
0192 394 : OUTPUTS:
0192 395 :
0192 396 : THE DISK'S SIZE, IN LOGICAL BLOCKS, IS OBTAINED FROM THE UCB
0192 397 : AND THE I/O IS COMPLETED WITH THE SECOND I/O STATUS LONGWORD
0192 398 : EQUAL TO THE DISK'S SIZE.
0192 399 :-
0192 400
0192 401 IOC$SENSEDISK::
0192 402 MOVL UCB$L_MAXBLOCK(R5),R1 ;SENSE DISK'S SIZE
0197 403 MOVZWL S^#SS$ NORMAL,R0 ;GET DISK'S SIZE IN LOGICAL BLOCKS
019A 404 BRW EXE$FINISHIO ;SET NORMAL COMPLETION STATUS
019D 405 ;FINISH I/O OPERATION
019D 406 .END
```

51 00B0 C5 D0
50 01 3C
FE63' 31

IOSUBRAMS
Symbol table

B 11
- NONPAGED RANDOM ACCESS MASS STORAGE I/ 16-SEP-1984 00:25:18 VAX/VMS Macro V04-00
5-SEP-1984 03:43:47 [SYS.SRC]IOSUBRAMS.MAR;1

Page 11
(6)

BUG\$ STRNOTWCB	*****	X	02
DEV\$ SQD	= 00000005		
DYN\$C-WCB	= 00000012		
EXE\$FINISHIO	*****	X	02
FCB\$ FILESIZE	= 00000038		
IOCS\$APPLYECC	00000000	RG	02
IOCS\$CVTLOGPHY	00000074	RG	02
IOCS\$CVTLOGPHYU	0000007D	RG	02
IOCS\$MAPVBLK	0000009F	RG	02
IOCS\$PTETOPFN	*****	X	02
IOCS\$SENSEDISK	00000192	RG	02
IOCS\$UPDATRANSP	00000156	RG	02
IRP\$ MEDIA	= 00000038		
MMG\$GC_SPTBASE	*****	X	02
PM\$SGL_HIT	*****	X	02
PR\$ IPC	= 00000012		
PR\$ TBIS	= 0000003A		
PTE\$ PFN	= 00000015		
PTE\$V PFN	= 00000000		
RVT\$ UCBLST	= 00000044		
SS\$ NORMAL	= 00000001		
UCB\$B FIPL	= 0000000B		
UCB\$ DEVCHAR	= 00000038		
UCB\$ DEVDEPEND	= 00000044		
UCB\$ MAXBLOCK	= 000000B0		
UCB\$ SVAPTE	= 00000078		
UCB\$ SVPN	= 00000074		
UCB\$M ECC	= 00000001		
UCB\$V NOCNVRT	= 00000002		
UCB\$W BCNT	= 0000007E		
UCB\$W BOFF	= 0000007C		
UCB\$W DA	= 000000BC		
UCB\$W DC	= 000000BE		
UCB\$W DEVSTS	= 00000068		
UCB\$W EC1	= 000000C4		
UCB\$W EC2	= 000000C6		
VAS\$ BYTE	= 00000009		
VAS\$V BYTE	= 00000000		
VAS\$ SYSTEM	= 0000001F		
WCB\$B ACCESS	= 0000000B		
WCB\$B TYPE	= 0000000A		
WCB\$L FCB	= 00000018		
WCB\$L LINK	= 00000020		
WCB\$L ORGUCB	= 00000010		
WCB\$L RVT	= 0000001C		
WCB\$L STVBN	= 0000002C		
WCB\$V NOTFCP	= 00000002		
WCB\$W NMAP	= 00000016		

! Psect synopsis !

PSECT name	Allocation		PSECT No.	Attributes													
. ABS .	00000000 (	0.)	00 (	0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE		
\$AB\$\$	00000000 (	0.)	01 (	1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE		

IOSUBRAMS
Psect synopsis

C 11
- NONPAGED RANDOM ACCESS MASS STORAGE I/ 16-SEP-1984 00:25:18 VAX/VMS Macro V04-00 Page 12
5-SEP-1984 03:43:47 [SYS.SRC]IOSUBRAMS.MAR;1 (6)

WIONONPAGED

0000019D (413.) 02 (2.) NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	29	00:00:00.07	00:00:00.96
Command processing	120	00:00:00.53	00:00:07.69
Pass 1	372	00:00:12.88	00:00:54.55
Symbol table sort	0	00:00:02.17	00:00:08.04
Pass 2	91	00:00:02.37	00:00:08.01
Symbol table output	7	00:00:00.08	00:00:00.08
Psect synopsis output	2	00:00:00.03	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	623	00:00:18.13	00:01:19.46

The working set limit was 1500 pages.
74962 bytes (147 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1452 non-local and 21 local symbols.
406 source lines were read in Pass 1, producing 14 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	13
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	19

1583 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:IOSUBRAMS/OBJ=OBJ\$:IOSUBRAMS MSRC\$:IOSUBRAMS/UPDATE=(ENH\$:IOSUBRAMS)+EXECML\$/LIB

0376 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

